ISSN : 2302-3805

IMPLEMENTASI DETEKSI OUTLIER PADA ALGORITMA HIERARCHICAL CLUSTERING

Ahmad Saikhu[2], Yoga Bhagawad Gita[2]

Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember (ITS), Kampus ITS Sukolilo Surabaya, 60111
[1]saikhu@its-sby.edu, [2]yoga_bg@cs.its.ac.id

Abstrak
Clustering memiliki permasalahan terhadap hasil yang dicapai apakah sudah sesuai dengan kebutuhan. Salah satu pilihan adalah agglomertive hierarchical clustering(AHC). Pada AHC ada dua hal penting yaitu pemilihan jenis pengukuran similarity dan teknik linkage. Pada beberapa kasus, penggunaan AHC yang berbasis single linkage menunjukkan hasil yang baik pada berbagai himpunan dataset dengan variasi jumlah dan bentuk. Namun, algoritma ini masih dapat menimbulkan masalah kepekaan pada adanya outlier dan fluktuasi kepadatan data. Juga tidak memungkinkan untuk menentukan jumlah klaster secara otomatis. Untuk mengatasi ini, diperlukan pendekatan untuk mendeteksi outlier dan secara otomatis menentukan jumlah klaster yang dibutuhkan secara tepat. Pendekatan ini merupakan pengembangan dari AHC yang dapat mendeteksi adanya outlier. Hasil uji coba menunjukkan bahwa modifikasi HCA lebih unggul dalam penanganan outlier.

Kata kunci:
Clustering, Single linkage, outlier.

1. Pendahuluan
Beberapa teknik *data mining* dapat dibagi atas teknik klasifikasi, *clustering*, penggalian kaidah asosiasi, analisa pola sekuensial, prediksi, visualisasi data dan lain sebagainya.Terdapat dua kategori yang bertugas untuk mengelompokkan data sehingga dapat diketahui keterkaitan antar satu data dengan data lainnya. Kategori tersebut adalah klasifikasi yang bersifat terawasi dan *clustering* yang bersifat tidak terawasi.
Algoritma *clustering* dapat dibedakan menurut cara kerjanya [1], yaitu:

a. *Partitional clustering*, contohnya adalah K-Means dan variasinya.
b. *Hierarchical clustering*, contohnya adalah *Agglomerative* dan *Divisive Hierarchical Clustering* dan viariasinya.

Pada algoritma *hierarchical clustering* terdapat beberapa keunggulan yaitu tidak perlu menentukan jumlah klaster yang diinginkan karena proses dapat langsung dihentikan pada saat jumlah klaster sesuai dengan yang diinginkan. Namun algoritma ini juga memiliki kelemahan bergantung pada pemilihan teknik *intercluster similarity* yang lebih dikenal dengan istilah *linkage*. Terdapat lima buah teknik *linkage* dasar. Beberapa kelemahan untuk teknik di atas adalah sensitif terhadap adanya *outlier*, kesulitan menangani variasi bentuk dan ukuran, dan memisahkan klaster yang besar.
Dalam jurnalnya, (Almeida, Barbosa, Pais, & Formosinho, 2007) memperkenalkan algoritma *clustering* berbasis hierarki yng menawarkan beberapa kelebihan, diantaranya tidak terpengaruh adanya *outlier* dan dapat mendeteksi jumlah klaster yang tepat. Untuk dapat mengetahui performa dari algoritma tersebut, maka dalam penelitian ini diaplikasikan dan dibandingkan dengan algoritma lainnya yang juga berbasiskan pada *hierarchical clustering* yaitu CURE [2].

2. Tinjauan Pustaka
Hierarchical clustering adalah metode yang digunakan untuk menemukan struktur yang mendasari obyek melalui proses iterasi yang mengasosiasikan (dengan *agglomerative*) dan memisahkan (dengan *divisive*) antara obyek dengan obyek dan berhenti ketika semua obyek telah diproses. *AHC* diawali dengan masing-masing obyek berada pada klaster yang berbeda kemudian dilakukan penggabungan klaster secara sekuensial, mengurangi jumlah klaster sampai semua obyek berada pada satu klaster. AHC divisualisasikan dalam bentuk dendrogram.

2.1. Pengembangan Hierarchical Clustering
Pada makalah ini diimplementasikan algoritma pengembangan dari algoritma *AHC* dengan tiga tahap, yaitu penghilangan *outlier*, pengelompokan klaster, dan pengklasifikasian *outlier* [3].
Penghilangan outlier merupakan tahap awal yang harus dilakukan. Tahap ini dilakukan agar proses pengelompokan klaster hanya terfokus dengan data asli tanpa terpengaruh oleh kehadiran data *outlier*. *Pseudocode* untuk tahap ini adalah:

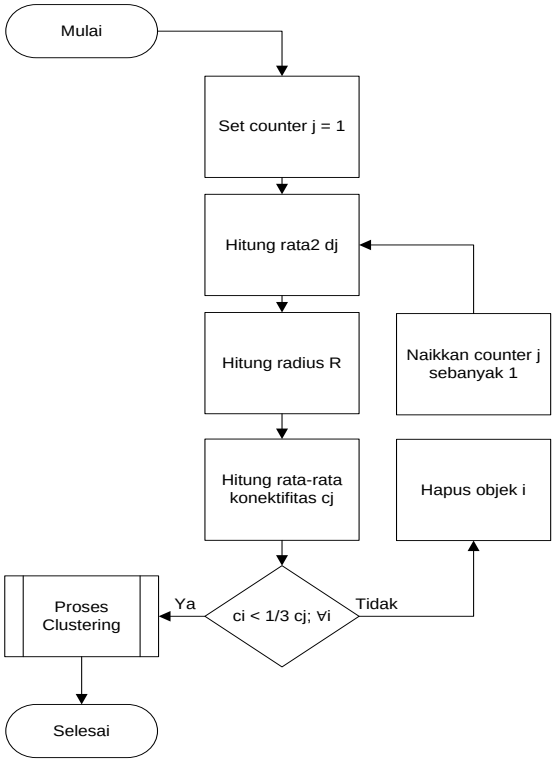
Set iteration counter $j = 1$
Repeat until number of discarded objects = 0
  Calculated $\bar{d}_j$
  Set $R = 4\bar{d}_j$
  Calculate $\bar{c}_j$ (R)
  Discard objects i if $c_i < 1/3\bar{c}_j$ (R)
  Increase j
End

ISSN : 2302-3805

/* followed by a new process in which $R = 2\bar{d}_j$ */

Gambar 1 Pseudocode fungsi hapus outlier

Nilai jarak dj adalah nilai rata-rata jarak terdekat yang dimiliki tiap objek terhadap objek lain yang dan dirata-rata dengan nilai dj yang ada pada iterasi sebelumnya. Nilai dj digunakan untuk menentukan nilai radius R di setiap iterasi.
Nilai koniktivitas ci adalah jumlah tetangga suatu objek i yang berada didalam nilai radius R. Nilai ci ini digunakan untuk menentukan pakah objek i merupakan outlier atau bukan dengan cara dibandingkan dengan nilai batas atau threshold. Nilai threshold ini didapat dari nilai cj dikali dengan 1/3, dimana nilai cj ini merupakan nilai rata-rata konektivitas semua objek yang bukan outlier pada iterasi tersebut.
Proses penghapusan outlier ini dilakukan berulang-ulang pada proses iterasi yang sama, yaitu menghitung dj, R, ci, dan cj untuk masing-masing iterasi. Pada akhir iterasi dilakukan proses pembuangan objek yang dianggap outlier.



Gambar 2 Flowchart penghapusan outlier

Tahap pengelompokan klaster adalah sebuah proses *clustering* akan otomatis menyesuaikan jumlah klaster yang dibutuhkan dengan berdasar pada pola data inputannya. Tahap ini memanfaatkan nilai-nilai hasil *linkage step* untuk mendapatkan variabel baru yang disebut *descriptive function* (DF).

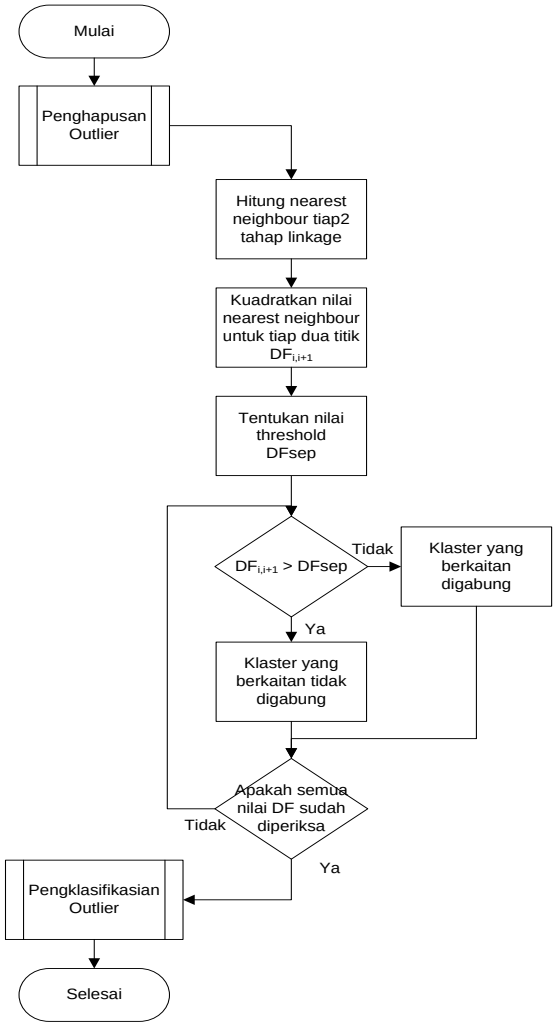
$$DF_{i,i+1} = d^2_{i,i+1} \quad (1)$$

Nilai di adalah jarak terpendek antara dua titik yang merupakan hasil perhitungan pada*linkage step* ke i. Nilai di,i+1 berbeda dengan nilai di, nilai di,i+1 adalah nilai minimal dari nilai-nilai di yang melibatkan dua buah objek pasangan objek yang memiliki variasi linkage step yang unik.

Dengan memanfaatkan semua nilai DF, didapatkan nilai *threshold* yang cocok untuk menentukan jumlah klaster yang tepat untuk sebuah dataset yang disebut DF separator.

$$DF_{sep} = 6 \times (Q_3 - Q_1) \quad (2)$$

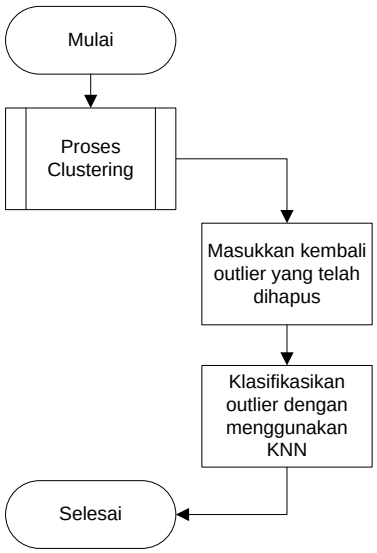
Nilai Q_1 dan Q_3 masing-masing adalah batas atas nilai kuartil pertama dan ketiga dari distribusi nilai pada *descriptive function*. Jika terdapat nilai DF yang lebih besar dari nilai DFsep maka *linkage step* yang memiliki nilai DF tersebut tidak akan dilakukan, atau dengan kata lain akan menjadi klaster yang terpisah.



Gambar 3 Flowchart proses clustering

ISSN : 2302-3805

Tahap terakhir adalah pengklasifikasian *outlier* yang telah dihilangkan pada tahap pertama. Objek-objek yang sudah dibuang pada tahap pertama merupakan objek yang dianggap outlier, sehingga untuk proses *clustering* yang lebih baik, objek-objek ini tidak diikutsertakan ke dalam proses *clustering*. Namun objek-objek ini tidak selalu berupa outlier. Objek-objek ini mungkin terkena efek dari fenomena 'dilusi' yaitu objek-objek yang berada pada batas klaster memiliki sebaran yang lebih besar jika dibandingkan dengan objek-objek yang berada di sekitar pusat klaster.



Gambar 4 Flowchart pengklasifikasian outlier

Setelah klaster terbentuk maka data yang dianggap sebagai *outlier* pada tahap pertama dimasukkan kembali dengan mengklasifikasikan *outlier* tersebut pada beberapa kelompok klaster yang telah terbentuk pada tahap kedua. Metode yang digunakan pada tahap ini adalah metode *supervised clustering* sederhana yaitu dengan menggunakan *k-nearest neighbour* (KNN).

Pada metode KNN yang digunakan, nilai k=1 adalah metode yang sederhana namun dapat diandalkan karena kelemahan pada 1NN yaitu sensitif terhadap adanya *outlier* tidak berpengaruh pada kasus ini. Secara garis besar flowchart dari algoritma ini ditunjukkan pada Gambar 2, 3., dan 4.

2.2 CURE

Pada penelitian juga diimplementasikan algoritma CURE (*clustering using representative*). Algoritma CURE ini dijadikan sebagai pembanding untuk mengetahui tingkat keberhasilan pada algoritma pengembangan dari *hierarchical clustering*.

CURE tidak menggunakan seluruh obyek yang ada untuk mendapatkan hasil linkage step yang akan digunakan untuk menggabungkan dua buah klaster, tetapi hanya menggunakan *representative point* yang jumlahnya tidak terlalu banyak sehingga dapat mempercepat waktu pemrosesan.

Representative point dari CURE ini hanya digunakan untuk acuan untuk melaksanakan penggabungan klaster, sedangkan objek klaster yang digabung adalah objek klaster yang asli yang merupakan keseluruhan objek.

Untuk menyimpan data yang ada dibutuhkan dua macam struktur data, yaitu kd-tree [4] [5] dan *heap* [6]. Kd-tree digunakan untuk menyimpan data *representative point*. Dengan menggunakan kd-tree, maka pencarian *nearest neighbour* dari suatu titik tidak perlu dihitung pada setiap data yang ada. Sedangkan heap digunakan untuk menyimpan data-data mengenai klaster dan juga diurutkan berdasarkan jarak ke klaster lain. Hal ini dilakukan untuk mempercepat proses pemilihan penggabungan klaster pada *linkage step*.

Setiap terjadi perubahan klaster perlu dilakukan perhitungan ulang untuk mendapatkan *representative ponit* yang baru. *Representative point* ini dipilih sedemikian hingga dapat merepresentasikan bentuk dari sebuah klaster. Untuk tujuan tersebut, maka bagian yang dipilih untuk menjadi *representative point* adalah pusat klaster dan batas-batas klaster. Untuk mencegah agar tidak memilih outlier sebagai *representative point* maka diperlukan sebuah nilai shrinking point α. Nilai α ini memiliki rentang 0 sampai 1 yang digunakan sebagai pengali jarak pusat ke batas klaster. Dengan demikian sensitifitas terhadap *outlier* akan berkurang.

$$w.rep = w.rep \cup \left(p + \alpha \times (w.mean - p)\right) \quad (3)$$

Variabel w.rep berisi himpunan *representative point* dari klaster w, sedangkan w.mean berisi nilai rata-rata dari klaster w. Nilai p adalah nilai sebuah objek dalam klaster w yang terpilih untuk dijadikan sebagai *representative point*. Nilai α adalah nilai *shrinking point*. Secara garis besar, *pseudocode* dari algoritma Cure ini dijabarkan pada Gambar 5 dan Gambar 6.

Prosedur klaster merupakan *pseudocode* untuk fungsi utama dari CURE, sedangkan prosedur merge merupakan *pseudocode* untuk fungsi penggabungan dua klaster dan menentukan *representative point*-nya.

3. Cluster Validation

Cluster validation adalah cara untuk mengetahui seberapa baik hasil yang diperoleh dari proses *clustering*. Jika pada klasifikasi pengukuran keberhasilan suatu proses klasifikasi dapat diketahui dengan jelas dengan melihat nilai presisi, *recall*, dan akurasinya yang bersifat eksak atau pasti. Hasil C*lustering* hanya bisa dietahui tingkat performanya dengan membandingkan hasil dari algoritma yang lain karena hasil *cluster validation* memiliki nilai yang tidak eksak.

ISSN : 2302-3805

```
procedure cluster(S, k)
begin
1.  T := build_kd_tree(S)
2.  Q := build_heap(S)
3.  while size(Q) > k do {
4.     u := extract_min(Q)
5.     v := u.closest
6.     delete(Q, v)
7.     w := merge(u, v)
8.     delete_rep(T, u); delete_rep(T, v); insert_rep(T, w)
9.     w.closest := x /* x is an arbitrary cluster in Q */
10.    for each x ∈ Q do {
11.       if dist(w, x) < dist(w, w.closest)
12.          w.closest := x
13.       if x.closest is either u or v {
14.          if dist(x, x.closest) < dist(x, w)
15.             x.closest := closest_cluster(T, x, dist(x, w))
16.          else
17.             x.closest := w
18.          relocate(Q, x)
19.       }
20.       else if dist(x, x.closest) > dist(x, w) {
21.          x.closest := w
22.          relocate(Q, x)
23.       }
24.    }
25.    insert(Q, w)
26. }
end
```

Gambar 5 Pseudocode fungsi utama CURE

```
procedure merge(u, v)
begin
1.  w := u ∪ v
2.  w.mean := (|u|u.mean+|v|v.mean) / (|u|+|v|)
3.  tmpSet := ∅
4.  for i := 1 to c do {
5.     maxDist := 0
6.     foreach point p in cluster w do {
7.        if i = 1
8.           minDist := dist(p, w.mean)
9.        else
10.          minDist := min{dist(p, q) : q ∈ tmpSet}
11.       if (minDist ≥ maxDist){
12.          maxDist := minDist
13.          maxPoint := p
14.       }
15.    }
16.    tmpSet := tmpSet ∪ {maxPoint}
17. }
18. foreach point p in tmpSet do
19.    w.rep := w.rep ∪ {p + α*(w.mean-p) }
20. return w
end
```

Gambar 6 Fungsi penggabungan pada CURE

Pengukuran hasil dari proses *clustering* memiliki banyak metode seperti pengukuran berdasarkan nilai korelasinya, nilai *cohession and separation*, *silhouette coefficient* (1), *Dunn index*, dan *Davies-Bouldin index* [7].

3.1 Korelasi

Perhitungan nilai korelasi tidak perlu dilakukan terhadap semua anggota matriks melainkan hanya n (n - 1) / 2 anggota, hal ini karena nilai *proximity matrix* dan *incidence matrix* merupakan matriks simetris. Nilai korelasi yang bagus adalah ketika menghasilkan jarak intraklaster yang kecil dan interklaster yang besar. Namun nilai pada *incidence matrix* berlaku sebaliknya.

3.2 Cohession and Separation

Nilai *cohession* merupakan nilai varian untuk mengukur seberapa dekat data-data yang berada pada klaster yang sama. Sedangkan *separation* adalah nilai varian untuk mengukur seberapa besar perbedaan data-data yang ada pada klaster yang berbeda.

Rumus *cohession* atau WSS (*within cluster sum of squares*) dan rumus *separation* atau BSS (*between cluster sum of squares*) dapat dijabarkan sebagai berikut.

$$WSS = \sum_{i=1}^{k} \sum_{x \in C_i} (x - m_i)^2 \quad (4)$$

$$BSS = \sum_{i=1}^{k} |C_i| (m - m_i)^2 \quad (5)$$

Nilai k adalah jumlah klaster yang terbentuk. Nilai $|C_i|$ adalah jumlah objek yang ada pada klaster i. Nilai m_i adalah rata-rata dari klaster i. Nilai m adalah mean dari keseluruhan data. Karena nilai cohession digunakan untuk mengukur nilai varian intraklaster maka hasil proses *clustering* akan semakin bagus jika nilainya semakin kecil. Pada separation berlaku sebaliknya karena merupakan hasil pengukuran varian pada interklaster.

3.3 Silhouette Coefficient

Silhouette coefficient merupakan validasi yang menggabungkan unsur *cohession* dan *separation*. Nilai dari *silhouette coefficient* ini memiliki rentang dari -1 sampai 1. Nilai 1 menandakan bahwa klaster yang terbentuk merupakan hasil klaster yang baik, sedangkan nilai -1 menunjukkan bahwa hasil klaster yang terbentuk adalah buruk.

$$s = \begin{cases} 1 - \frac{a}{b}, jika\ a < b \\ \frac{b}{a} - 1, jika\ a \geq b \end{cases} \quad (6)$$

Nilai a adalah rata-rata dari jarak objek i ke objek-objek lain yang berada pada satu klaster. Sedangkan nilai b adalah minimal dari rata-rata jarak objek i ke objek-objek lain yang berada pada klaster yang berbeda.

3.4 Dunn Index

Dunn index adalah salah satu pengukuran *cluster validity* yang diajukan oleh J.C. Dunn.*Cluster validity* ini berlandaskan pada fakta bahwa klaster yang terpisah itu biasanya memiliki jarak antar klaster yang besar dan diameter intra klaster yang kecil.

$$D = \min_{i=1 \ldots n_c} \left(\min_{j=i+1 \ldots n_c} \left(\frac{d(c_i, c_j)}{\max_{k=1 \ldots n_c} (diam(c_k))} \right) \right) \quad (7)$$

Dimana nilai $d(c_i, c_j)$ dan $diam(c_i)$ ini didefinisikan sebagai berikut.

ISSN : 2302-3805

$$d(c_i, c_j) = \min_{\substack{x \in c_i \\ y \in c_j}} (d(x, y)) \quad (8)$$

$$diam(c_i) = \max_{x,y \in c_i} (d(x, y)) \quad (9)$$

Nilai pada Dunn index ini jika semakin besar, maka hasil *clustering* akan semakin bagus.

3.4 Davies-Bouldin Index

Davies-Bouldin index merupakan *cluster validity* yang dibuat oleh D.L. Davies dan D.W. Bouldin yang memiliki fokus pada *similarity* antar klaster. Nilai *similarity* (R) ini berbasiskan pada nilai penyebaran dalam klaster (s) dan nilai *dissimilarity* antar klaster (d).

$$DB = \frac{1}{n_c} \sum_{i=1}^{n_c} \left(\max_{\substack{j=1..n_c \\ i \neq j}} (R_{ij}) \right) \quad (10)$$

Dimana nilai *similarity* R_{ij} didefinisikan sebagai berikut.

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \quad (11)$$

$$s_i = \frac{1}{\|c_i\|} \sum_{x \in c_i} d(x, v_i) \quad (12)$$

$$d_{ij} = d(v_i, v_j) \quad (13)$$

Nilai v_i adalah nilai dari pusat klaster ke i.

4. Hasil dan pembahasan

Pada uji coba ini terdapat lima macam skenario yang akan diuji coba.

- Uji coba bentuk
  Uji coba ini dilakukan untuk mengetahui kehandalan algoritma dalam melakukan *clustering* data pada bentuk-bentuk data yang sederhana sampai yang kompleks.
- Uji coba distribusi data
  Uji coba untuk mengetahui pengaruh perbedaan distribusi data pada hasil *clustering*.
- Uji coba kepadatan data
  Uji coba dilakukan untuk mengetahui apakah algoritma *clustering* mampu menangani data dengan kepadatan yang berbeda-beda.
- Uji coba jenis outlier
  Algoritma *clustering* akan diuji dalam penanganan outlier
- Uji coba data nonsintesis
  Algoritma *clustering* akan diuji pada data nonsintesis yang bukan data hasil generator.

Pada masing-masing skenario uji coba dilakukan satu sampai lima macam uji coba dengan menggunakan dataset yang berbeda. Masing-masing *cluster validation* menghasilkan kesimpulan untuk memilih algoritma mana yang lebih unggul.

Jika suatu algoritma memiliki total keunggulan yang lebih banyak atau dominan dibandingkan dengan algoritma yang lain, maka algoritma tersebut dinyatakan lebih unggul. Dari hasil uji coba pada semua skenario uji coba, didapatkan hasil yang dijabarkan pada tabel 1. Pada hasil tersebut terlihat bahwa pada skenario uji coba yang pertama mengenai bentuk klaster diperoleh hasil bahwa algoritma CURE yang lebih baik. Hal ini karena pada algoritma modifikasi HCA memiliki kinerja seperti pada *single linkage hierarchical clustering* yang lebih mengutamakan tetangga terdekat. Namun *cluster validity* yang digunakan sebagian besar memiliki karakteristik yang mengutamakan jarak intraklaster yang kecil dan jarak interklaster yang besar dimana hanya sesuai pada bentuk klaster yang sederhana, sedangkan dalam skenario uji coba yang pertama digunakan dataset dengan bentuk yang rumit.

Tabel 1 Rangkuman hasil uji coba skenario 1-5

Skenario	Jenis Uji Coba	Algoritma yang Unggul
1	Bentuk klaster	CURE
2	Distribusi data	CURE
3	Kepadatan data	Seimbang
4	Jenis outlier	Modifikasi HCA
5	Data nonsintesis	CURE

Jika dalam dataset tidak terdapat outlier, maka algoritma modifikasi HCA akan tetap memaksa untuk mengidentifikasi outlier sehingga menyebabkan hasil yang didapatkan menjadi kurang optimal. Kesimpulan ini berlaku juga untuk skenario uji coba yang kelima, yaitu pada data nonsintesis.

5. Kesimpulan

Berdasarkan hasil uji coba, maka didapat beberpa kesimpulan sebagai berikut:

- Algoritma modifikasi HCA lebih unggul dalam penanganan outlier.
- Algoritma modifikasi HCA kurang efektif dalam menangani data yang tidak mengandung outlier.
- Algoritma modifikasi HCA tidak memerlukan parameter-parameter seperti jumlah klaster yang dibutuhkan karena adanya otomatisasi pendeteksiannya.

6. Daftar Pustaka

[1] Tang, Pang-Ning, Steinbach, Michael dan Kumar, Vipin, 2006, CSE User Home Pages. *Introduction To Data Mining*. [Dikutip:15 Oktober 2010] http://www-users.cs.umn.edu /~kumar/dmbook/index.php.

ISSN : 2302-3805

[2] Guha, Sudipto, Rastogi, Rajeev dan Shim, Kyuseok, 1998, CURE: An Efficient Clustering Algorithm for Large Databases. Stanford : SIGMOD, Vol. 27.

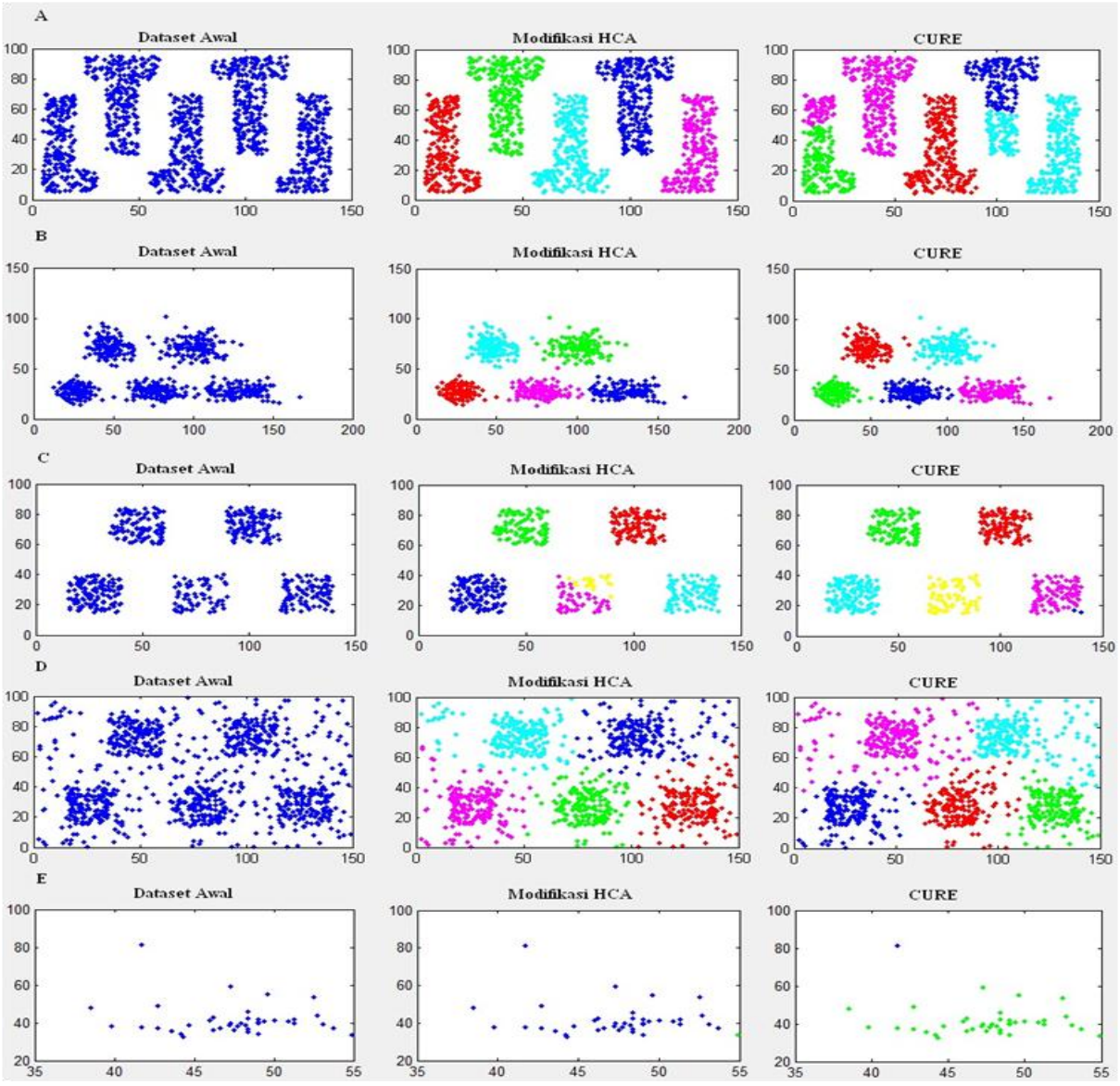
[3] Almeida, J.A.S., et al., 2007, Improving hierarchical cluster analysis: A new method with outlier detection and automatic clustering. Coimbra : Science Direct, Chemometrics and Intelligent Laboratory Systems, hal. 208-217.

[4] Wikipedia. k-d tree. *Wikipedia.* [Online] Wikipedia, 2011. [Dikutip: 10 Februari 2011.] http://en.wikipedia.org/wiki/K-d_tree.

[5] Chandran, Sharat, 2002, *Data Structures: Spring*. *Department of Computer Science.* [Online] 2002. [Dikutip: 22 Maret 2011.] http://www.cs.umd.edu/class/spring2002/cmsc420-0401/.

[6] Cormen, Thomas H., et al, 2001, *Introduction to Algorithms, Second Edition.* s.l. : The MIT Press, 2001. ISBN:0262032937.

[7] Kovacs, Ferenc, Legany, Csaba dan Babos, Attila, 2005, *Cluster Validity Measurement Techniques.* Budapest : Department of Automation and Applied Informatics Budapest University of Technology and Economics.



Gambar 7 Hasil uji coba pada (A) skenario pertama, (B) skenario kedua, (C) skenario ketiga, (D) skenario keempat, dan (E) skenario kelima